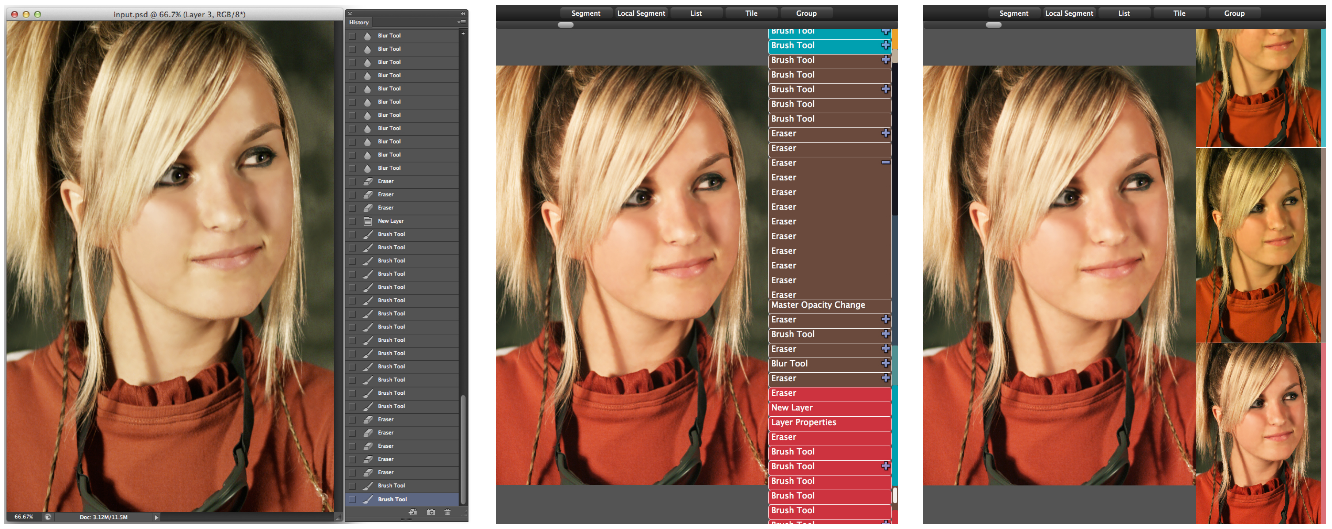
Data-driven Adaptive History for Image Editing

Hsiang-Ting Chen
University of Technology Sydney

Li-Yi Wei
University of Hong Kong

Björn Hartmann
UC Berkeley EECS

Maneesh Agrawala
Stanford University



(a) *Photoshop history list* **(b)** *Our history with semantic segments* **(c)** *Our semantic navigation*

Figure 1: *User interface of our system. (a) is the Photoshop history list. (b) is our data-driven adaptive history list. Our UI assigns a unique color to all entries belonging to the same semantic segment, with the scroll bar (right) colored accordingly. Our UI also groups repeating commands into tiles, which can be folded/unfolded via the corresponding plus/minus signs. (c) is the thumbnail preview mode for semantic segments. The color bar at right shows the assigned colors for the segments as in (b). Double clicking a thumbnail image would lead users to the corresponding segment in the history list. (Note: all screen shots display the same image; the perceived color differences are caused by different color spaces of Photoshop and our Qt-based UI.) Please refer to the accompanying video for live actions. (Photo credit: Kevin Dooley)*

Abstract

Digital image editing is usually an iterative process; users repetitively perform short sequences of operations, as well as undo and redo using history navigation tools. In our collected data, undo, redo and navigation constitute about 9 percent of the total commands and consume a significant amount of user time. Unfortunately, such activities also tend to be tedious and frustrating, especially for complex projects.

We address this crucial issue by *adaptive history*, a UI mechanism that groups relevant operations together to reduce user workloads. Such grouping can occur at various history granularities. We present two that have been found to be most useful. On a fine level, we group repeating commands patterns together to facilitate *smart undo*. On a coarse level, we segment commands history into chunks for *semantic navigation*. The main advantages of our approach are that it is intuitive to use and easy to integrate into any existing tools with text-based history lists. Unlike prior methods that are predominately rule based, our approach is data driven, and thus adapts better to common editing tasks which exhibit sufficient diversity and complexity that may defy predetermined rules or procedures.

A user study showed that our system performs quantitatively better than two other baselines, and the participants also gave positive qualitative feedbacks on the system features.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces—Graphical user interfaces (GUI);

Keywords: image editing, interaction, adaptive history, smart undo, semantic navigation

1 Introduction

Undo, redo, and history navigation are the most commonly used operations in interactive image editing [Lafreniere et al. 2010; Myers et al. 2015]. In our data collected from digital artists, such meta commands constitute about 9% of all issued commands (66 out of 797 per task average). However, they are also the most tedious, as it may take many repetitive operations to reach the intended points on the editing histories. This can greatly diminish user efficiency and satisfaction in common image editing tasks.

I3D 2016, February 27-28, 2016, Redmond, WA
ISBN: 978-1-4503-4043-4/16/03
DOI: http://dx.doi.org/10.1145/2856400.2856417

Prior works attempted to address this critical issue by proposing new graphical history representations or better undo mechanisms. However, these methods either introduce completely different representations from the traditional linear histories and thus present usability issues (e.g. hierarchical graphs in [Chen et al. 2011]) or hardcoded rules that might not be flexible enough in handling more complex real-world cases (e.g. regular expressions in [Denning et al. 2011] and spatial filters in [Meng et al. 1998]). Thus, a linear text list, despite its deficiencies, remains the dominant form of operation history representation in image editing software.

We propose *adaptive history*, a data-driven method to aggregate user editing operations for more efficient undo, redo, and history navigation (Figure 1).

Our two key observations are that 1) users often mentally chunk sequences of low-level editing operations into high-level, semantically meaningful units; during undo operations, users typically seek to undo an entire semantic unit instead of just individual operations, and 2) there are certain repeating command patterns with frequent occurrences in the command histories which are semantically inseparable (e.g. copy + paste or patch tool + patch selection in Photoshop). The challenge is to identify the boundaries between such semantic units. By collecting and analyzing actual data from professional artists through instrumented image editing tools, we use machine learning algorithms to train a classifier that can predict the boundaries between semantic units. We employ two particular machine learning methods for two levels of editing history granularity:

Semantic navigation On a coarse level, we use support vector machines (SVM) to identify high level semantic segments, such as skin smoothing or hair coloring for portrait retouching.

Smart undo On a fine level, we use n-gram analysis to identify operations often grouped together to accomplish specific micro tasks, such as "patch tool, patch selection, deselect" for blemish removal in Photoshop.

Overall, our approach belongs to the emerging trend of data-driven user interfaces (e.g. [Grabler et al. 2009; Su et al. 2009; Hartmann et al. 2010; Chaudhuri and Koltun 2010; Li et al. 2011; Lafreniere et al. 2011; Lu et al. 2012; Kazi et al. 2012; Yu et al. 2012; Limpaecher et al. 2013; Iarussi et al. 2013; Berger et al. 2013; Zitnick 2013; Pavel et al. 2013; Kazi et al. 2014b; Chen et al. 2014; Lu et al. 2014; Xing et al. 2014; Kazi et al. 2014a; Xing et al. 2015]), with specific applications in interactive image editing and history navigation. Our user interface design leverages the familiar UI concept of traditional linear history lists: for appearances, we enhance traditional linear history lists with visual cues with colors and foldable items; for behaviors, users still interact with our adaptive history lists as usual, but have the option to navigate through semantic chunks rather than just individual commands.

We evaluate our system by comparing it with traditional linear text history lists and time-based clustering of command history in [Li et al. 2011]. The result shows that our system performs significantly better than two other baselines. The participants, all of them professional artists, provided positive feedback on the proposed system features and commented that our system could be very helpful to their daily tasks.

2 Previous Work

Here we review prior art most relevant to our work.

2.1 Operation Sequence Analysis

Data analysis has been extensively applied for operation sequences of content creation tools. However, efforts so far have been primarily focusing on text editing, such as word processing [Kay and Thomas 1995; Linton and Schaefer 2000] and software development [Murphy et al. 2006]. Similar efforts for image editing have made significant progress recently, such as command vocabularies [Lafreniere et al. 2010], command recommendation [Li et al. 2011], UI customization [Lafreniere et al. 2011], command browsing [Pavel et al. 2013], and sequence prediction [Xing et al. 2014]. We encourage readers to see a recent survey on different conceptual models from Nancel and Cockburn [2014]. Such data driven analysis has yet to be applied for facilitating common undo, redo, and history navigation operations, which our work targets.

2.2 Graphical History

Graphical history visualization has long been an active research field; see a comprehensive suvey from [Heer et al. 2008]. Here, we focus on works most relevant to operation history in real-world settings (where each history may contain hundreds of commands). These focus on two main strategies: clustering and filtering.

The basic idea behinds *clustering* is to group similar or related commands together to reduce visual clutter [Kurlander and Feiner 1991; Nakamura and Igarashi 2008; Grabler et al. 2009]. Example methods include Chronicle from [Grossman et al. 2010] which builds the history hierarchy based on the user "save" commands, Meshflow from [Denning et al. 2011] which clusters operations based on hand-crafted regular expression rules, and [Chen et al. 2011] which visualizes editing histories via hierarchical graphs.

The basic idea behinds *filtering* is to selectively display information based on contextual criteria, including spatial relationships [Meng et al. 1998; Nakamura and Igarashi 2008; Denning et al. 2015; Chen et al. 2011], temporal relationships [Grossman et al. 2010; Denning et al. 2011], or content properties [Kurlander and Bier 1988; Callahan et al. 2006; Su 2007; Denning and Pellacini 2013; Chen et al. 2014].

A notable difference between these prior techniques and our method is that they are based on rules or heuristics whereas ours is data driven. Such hand crafted rules or heuristics are usually difficult to derive and have built-in assumptions that might not hold when facing the diversity real-world command histories. A data-driven approach could instead learn from such diversity and provide better results.

2.3 Undo Management

Navigating editing history is one of the most common operations for interactive editing tools, in particular undo and redo. For digital content creation, methods such as [Meng et al. 1998; Su 2007; Chen et al. 2011; Myers et al. 2015] provided the functionality of "undo by selection", and Edward et al. [2000] allowed users to bound atomic operations into semantic "chunks" for better navigation while also prevent accidental undo that breaks such chunk. Inspired by Edwards et al. [2000], the short-term "smart undo" function in our system also serves a similar purpose. However, instead of manually defined rules as in these prior works, we learn the operation patterns from analyzing collected user data. Thus, our algorithm is not constrained to any pre-defined rules and thus can adapt to different users, tasks, and tool versions.

3 User Interface

The history panel serves as a core UI component in common image editing systems such as Photoshop and GIMP, with main usages including undo, redo and history navigation. However, its basic representation and interaction have not evolved much from the original linear text-based lists. We propose a data-driven adaptive history with two key enhancements over traditional history lists: *semantic navigation* at a coarse level and *smart undo* at a fine level.

This two-level design was based on our informal discussions with artists as well as field observations on their workflows. In particular, we noticed that artists tend to work on two different levels of granularity:

- On a coarse level, they usually used layers or files to manage and store progress milestones, e.g. finished eye retouching or skin smoothing.
- On a fine level, they frequently undid through history list progresses that were considered to be unsatisfactory or exploratory.

Our *semantic navigation* and *smart undo* have been designed in response to these two levels of common artist workflows. Even though prior works have proposed fancier multi-resolution navigation (e.g. [Chen et al. 2011; Denning et al. 2011; Denning et al. 2015]), we believe this is not necessary, due to both prior user studies (e.g. "users tended to stay on a single resolution to avoid losing context of their navigations" as in [Chen et al. 2011]) as well as our own observations as described above.

From the users' perspective, our interface design minimally changes the feel and look of the traditional history lists. In particular, users still interact with our adaptive history as usual, but with enhanced interaction through semantic navigation and smart undo, and enhanced perception through color coding and foldable tile-based command grouping. From the developer's perspective our interface design, due to its minimal change from traditional history lists, should be straightforward to port into their existing UIs. Our underlying algorithms are all data-driven; with proper training data, developers can train their own classifier as either a fixed preprocess or as an ongoing process that dynamically adapts to user behaviors.

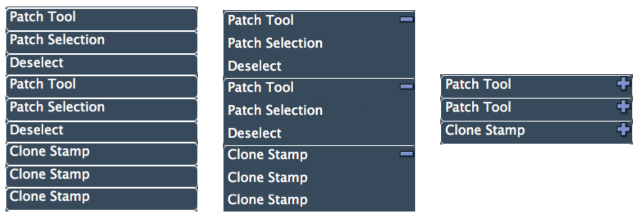
3.1 Semantic Navigation

Scenario Alex is in the middle of a photo retouching session. He has just finished the hair part of the portrait and was performing the skin smoothing. Feeling uncomfortable about the overall feeling of skin, he decided to undo all the skin smoothing work he has done and start over. At this point, as the history list has already grown too long and the image already contains too many layers, Alex had to go through lots of trials-and-errors to reach the right roll-back spot via the Photoshop history list (Figure 1a).

With our system, Alex can clearly see the automatically colored and aggregated segments of his work history (Figure 1b) and locate the right spot of interest. Our system also provides the *dual view* function that allows Alex to switch between text and thumbnail view (Figure 1c) and performs analogous operations.

Design We design our system with simplicity and minimal changes to traditional linear histories. The segments are visualized with different colors and correspond to semantic chunks such as hair retouch, skin smoothing, and eye sharpening. We pick the color scheme with the following goals in mind: 1) segment colors should differ sufficiently from each other for distinctiveness, and 2) the colors should be dark enough to visualize the text labels, which are all white instead of in different colors to avoid confusion. In our early design phase, we have tried to use colors based on the underlying color changes of the corresponding image regions. However, this might not produce distinctive enough colors. For example, the representative colors for retouching eyes and retouching hair can be very similar.

The segmentation is performed on the fly, based on the classifier trained from the data collected from the professional artists. Users can adjust the granularity of the segmentation using the slider bar at the top of the UI (Figure 1b). In addition, we also modify the color of the scroll bar beside the history list, so that users can see the relative lengths of the segments (Figure 1b).



(a) *lists* **(b)** *tiles* **(c)** *groups*

Figure 2: *History list display modes. Our history list has three display modes: list, tile, and group. We provide zoom-in views from our main UI in Figure 1 for easier comparison. List mode (a) is the basic text-based list with semantic colors, tile mode (b) groups the detected repeating commands into a single tile, and group mode (c) folds the command groups in (b) into single entries. Users can also manually fold/unfold by clicking on the plus/minus sign on each group tile.*

3.2 Smart Undo

Scenario During skin smoothing, Alex uses the healing patch tool to remove large area of undesired features on the skin. Each removal operation involves three steps: (1) select the target region to be retouched, (2) drag the target region to the artifact-free source region, and (3) deselect the target region. If he is not satisfied with the result, Alex would have to issue the undo command three times in a row to fully undo the retouching effect. Similar scenarios also happen on other tools such as the clone stamp tool, the pen tool (add anchor point, drag anchor point) and common action patterns such as copy-paste-move. Also as Alex is using the brush tool, the history list is quickly filled with identical "brush tool" entries. If he is not satisfied with the outcome, using the traditional interface he would have to continuously issue undo command and visually check the result to find the point he wishes to return to.

With our system (Figure 2), such repeating commands are automatically grouped into a foldable single entry. Alex can smart undo the whole chunk of command and navigate through the history list more easily.

Design We provide three different display modes for visualizing the repeating command patterns (Figure 2). Besides the basic command list, users can switch to "tile" mode, where repeating patterns are grouped into single tiles. In "group" mode, tiles are automatically folded into single entry, which users can manually fold/unfold by clicking on the plus/minus icons. In these modes, the smart undo function can undo a whole tile/group in addition to a single command.

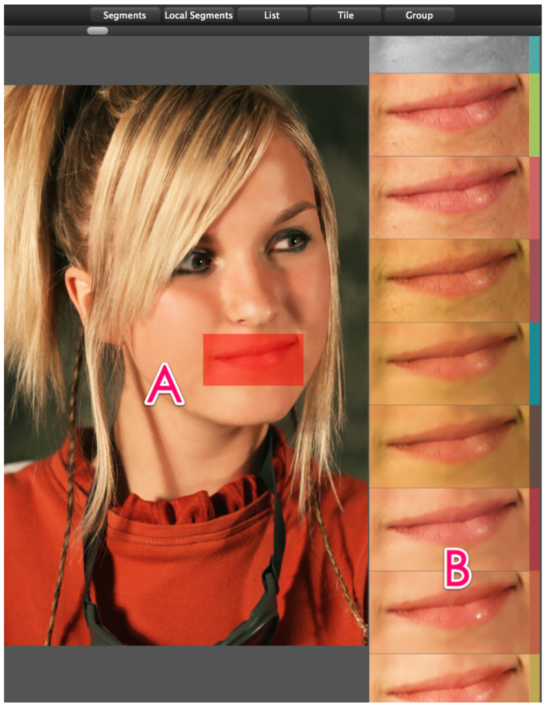


Figure 3: *Spatial filter. Users can select particular regions of interest (A), and review the corresponding history via the thumbnail images at (B). (Photo credit: Kevin Dooley)*

3.3 Spatial Filter

Besides two major functions mentioned above, we also provide a spatial filter function to enhance the usability of our system. Spatial filter allows users to select particular image regions of interest and see only the relevant portions of the command history. For example, Alex might want to roll back to the point before he retouched the eyes. With this function, he can select the region of interest and our system will show the thumbnail image of the region in each semantic segmentation (Figure 3).

4 Method

The main challenges for *semantic navigation* and *smart undo* are to segment the working history into semantically meaningful segments and to identify atomic repeating patterns. We adopt a data-driven approach based on two hypotheses:

H_1 Artists tend to use different command and/or parameter sets for different retouching tasks, e.g., hairs or lips.

H_2 Experienced artists tend to perform certain image editing procedure in a fixed order, e.g., copy-paste-move.

A valid H_1 implies a classifier that segments and labels the given editing history list. A valid H_2 implies the usefulness of the smart undo function.

In the following sections, we describe our data collection process, examining our hypotheses, followed by the algorithm details.

4.1 Data Collection

Instrumentation We chose Photoshop as our target software due to its popularity among digital artists. The instrumentation is achieved by the *history log tool* provided in Photoshop, which outputs commands and parameters in a human-readable text form, as well as a Photoshop scripts written by ourselves that records intermediate frames of every command in the history as well as stroke motions and parameters of brush-operations that were not captured by the Photoshop *history log tool*.

Artists To collect the training data, we recruited 10 freelancer artists on oDesk. They were all professional photo retouchers with at least 5 years of working experiences residing in different countries including Belarus, Bulgaria, Colombia, Philippines, Serbia, Ukraine and United States. The average hourly payment was 14.3 U.S. dollars ($\sigma = 5.4$). Six of them used keyboards and mice as their primary interaction devices while the other four used Wacom pen tablets.

Tasks We then gave each artist four portrait photos downloaded from Flickr for retouching. We were aware of the fact that some artists may prefer to perform minimal retouching to keep the portraits as natural as possible. To avoid such situation and make sure our collected data contains enough information, we gave artists a minimum retouch requirement list, e.g. you must retouch the hair, whiten the eyes, remove blemish, etc. Interestingly, in the end we noticed that all retouchers actually performed much more retouches than we asked. We believed this implied that these professional retouchers truly respected their works and had a very high bar on the quality of the portrait shot. For the retouching of all four photos, the average working time was 9 hours and 28 min ($\sigma = 1$ hour and 55 min) and the average number of collected operations per artist was 852 ($\sigma = 444$, $min = 893$, $max = 5977$).

After finishing retouching, we asked the artists to segment and label their own retouching processes into smaller sub-tasks based on the reference retouching tag lists we provided (Appendix A). This list was compiled with the chapter/section names of the photo retouching tutorial book [Kelby 2011] with entries like "eye sharpening", "blemish removal", "skin smooth", etc. We also made it clear to artists that they can add their own tag descriptions if necessary.

In summary, we collected three types of data: fine-grained operation sequences (in text), corresponding image content of each operation, and the text labels describing the operation sequences provided by the artists.

4.2 Insights from Collected Data

Hypothesis H_1 is false After examining the data, we surprisingly found out that it is difficult to statistically distinguish the command sets of different retouch tasks. For example, we expect to see dramatically different command sets between *eye sharpening* and *skin smoothing*. However, multiple artists used similar commands for both, e.g. even sharpen filter (with masked layers) appears in the *skin smoothing* tasks to compensate the blurred facial features.

However, the data shows that when transition from one image editing task to another, artists tend to create new layers, start using new editing operations, and adjust the parameters, such as brush size, brush color etc. Editing operations are also more likely to be applied at different spatial location during the transition. This observation leads to the use of a binary classifier (SVM) that identifies the transition spot between tasks and divides the editing history into semantic segments. Note that some meta operations, such as layer creation or layer merge, frequently appear at the boundaries of semantic segments. However, naively treat such layer operations as transition points may not produce the desired outcomes. For example, skin smoothing often involves the creation and combination of multiple layers, which should belong to one instead of multiple semantic segments.

Hypothesis H_2 is true As we hypothesized, the collected data contains many short repeating patterns with high frequency and are semantically inseparable. Common examples include “Patch Tool, Patch Selection, Deselect” for skin retouch, “Add Anchor Point, Drag Anchor Point” for path selection, and “Master Opacity Change, Merge Layer” for finishing retouching sub-tasks. The inseparability means that it is usually meaningless to undo or navigate to the middle of such chunks.

4.3 Semantic Segmentation

Per our insights from the collected data, we design a Support Vector Machine (SVM) classifier to label the transition spots between retouching tasks, i.e., the boundaries between semantic segments.

Feature Vector We associate a feature vector with each operation; the vector is defined as the combination of the operation histogram of the k preceding operations and the corresponding affected image regions. Note that we use a causal window so that we can perform on-the-fly prediction based on users’ current operation.

Formally, for the j_{th} operation, we denote its previous k operations as its neighbor $N(j)$ and its feature vector $F(j)$ can be written as:

$$F = \{O, P\} \tag{1}$$

where:

- $O \in R^o$ indicates the histogram of operations in N, R^o the o-dimensional (reduced) operation space (explained below), and $O[i]$ the number of type-i operations in N.
- $P \in R^p$ is the position vector; we divide the image into p grid cells (default $p = 16$), and $P_i = 1$ if the content at i_{th} cell is modified and 0 otherwise.

With the feature vector defined as above, we train the SVM classifier using LIBSVM [Chang and Lin 2011]. We found $k = 10$ yield reasonable precision and used it for all our experiments.

Operation space dimension reduction There are about 168 basic operations in Photoshop (command information obtained from Photoshop $\rightarrow$ Edit $\rightarrow$ Keyboard Shortcuts $\rightarrow$ Summarize). We found the large number of operations hamper the ability of our classifier to extract meaningful information from collected user editing sequences. To address this issue, we reduce the dimensionality of operation space, i.e., the dimensions of O in the feature vector.

Among all Photoshop operations, some have similar semantic meanings, such as (“New Layer”, “New Color Filled Layer”, “Layer from Background Color”) and (“Rotate 90 degree clockwise”, “Rotate 180 degree clockwise”), while others are designed for similar purposes, e.g., there are 15 different blurs in Photoshop. Fortunately, the Photoshop menu hierarchy already provides good suggestions about command clusters; for example, all blurs commands are clustered under the menu item of “blur”. This allows us to reduce the dimension of the command space from 168 to 41 by simply classifying all non-root-level commands into its ancestor command at the root level menu.

Validation To exam the precision of our SVM classifier, we performed a 4-fold validation. In our data set, each of 10 artists was asked to retouch and label four photos. We trained the classifier from 3 of the 4 photos (i.e. 30 photo retouch history in sum) and predict the remaining one (i.e. 10 photo retouch history). The average prediction precision is 96%.

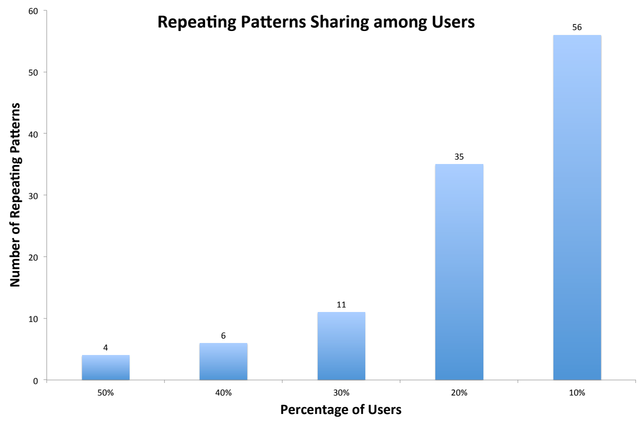


Figure 4: *Pattern usage chart. There are 4 patterns (average length 3) shared by half of the artists while 56 patterns (average length 3.7) used by only one artist.*

However, our data set is asymmetric where most positions are negative (not boundary) and only very few of them, which lie on segmentation boundaries, are positive. If we only look at the positive label, the prediction precision is around 70%. More specifically, around 80% of the errors are false positive, which causes over-segmentation of the given editing history.

We believe that the lower prediction rate is due to the different granularity of labeling in our data set. For example, some artists might label the whole chunk of operations as *face retouch* while others might separate it into *skin smoothing*, *blemish removal*, etc. As a result, applying a classifier with finer granularity to data with coarse labels will produce many false positive labels.

Fortunately, in our user study, such over segmented results did not turn out to negatively affect users’ experience of semantic navigation. Over-segmentation might decrease the effectiveness of semantic navigation, but it is still much better than traditional non-segmented history lists.

4.4 Smart Undo

N-gram analysis To extract the useful repetitions, we perform standard n-gram analysis ($n >= 3$). There are lots of redundant patterns in the n-gram table, e.g. the pattern of $ABCABC$ is the complete repetition of ABC, and we should only insert the ABC pattern into our pattern table. We resolve such redundancy by a top-down $\rightarrow$ bottom-up approach. We first scan the table from top-down (larger n to smaller n), and remove the patterns that are complete repetitions of some existing patterns (e.g., $ABCABC$ is the complete repetition of ABC). Then in second pass, we perform a bottom-up scan to remove patterns which are sub-strings of other longer patterns and has same count (e.g. ABC has 3 appearance, and there are already 2 $ABCE$ and 1 $ABCD$ in the table).

Repeating pattern By performing the n-gram analysis, we extract command patterns with length at least 3 and occur more than 3 times in the data. We found 112 such patterns with the average occurrence count of 7 (σ= 27).

Figure 4 shows that users usually have their own particular personal

Pattern	Count	User%
Eraser, Master Opacity Change, Eraser	26	50%
Move, Free Transform, Move	11	50%
Master Opacity Change, Hue/Saturation Layer, Modify Hue/Saturation Layer	8	50%
Brush Tool, New Layer, Brush Tool	7	50%
Patch Tool, Patch Selection, Deselect	295	40%
Blending Change, Master Opacity Change, Blending Change	12	40%

Table 1: *Pattern usage table. Here we show top patterns that are used by most artists.*

Pattern	Count	User%
Healing Brush, Layer Via Copy, Gaussian Blur, High Pass, Curves Layer, Modify Curves Layer, Group Layers, Blending Change, Channel Mixer Layer, Modify Channel Mixer Layer	4	10%
Duplicate Layer, High Pass, Gaussian Blur, Invert, Blending Change, Blending Options, Add Layer Mask, Invert, Brush Tool	4	10%
Duplicate Layer, High Pass, Blending Change, Add Layer Mask, Invert, Eraser	5	10%
Master Opacity Change, Duplicate Layer, Merge Layers, Selective Color Layer, Fill, Brush Tool	4	20%
Lasso, Paste, Layer Order, Move, Free Transform, Eraser	5	10%

Table 2: *Table with longer command patterns. There are some longer repeating patterns, usually reflection of preferences of individual or small groups of users.*

usage patterns but about half of them are still shared by two or more users. While Tables 1 and 2 show that shorter command patterns are more likely to be shared among users while longer ones usually reflects the personal usage patterns.

The command-patterns with high numbers of occurrence across users imply that they should be treated as an atomic operation in the history list and thus lead to our idea of “smart undo”. Such patterns can also help improve the design of image editing softwares or as macros to distribute over the community [Berthouzoz et al. 2011; Lafreniere et al. 2011].

Identical repetition It is common for users to issue identical commands repeatedly, such as brush, clone tool or move. Some previous works count all such repeating commands as single entries [Kurlander and Feiner 1992; Lafreniere et al. 2011], which we believe is not correct since identical command type could still have different semantic meanings (e.g. brush operations for retouching hair and retouching skin should not be clustered together). In our system, we group such identical repeating commands based on the position of modified region and time spent on operations. More specifically, we group commands together if they modify the same region of the image (based on the position vector) or is issued in less than 500 ms interval. Note that parameters of operations could also serve as a good factor for grouping, but currently it is not possible to obtain parameters related to brush operations in Photoshop due to the SDK constraint.

5 System Implementation

We built our system as a standalone application instead of part of Photoshop; the core algorithms are implemented with Qt and the user interface with QML. With newly announced Connection SDK from Adobe, our application communicates and synchronizes with Photoshop via TCP protocol.

It is certainly possible to implement our system via Photoshop plug-in for direct integration, allowing users to perform all the interaction and editing in one place. However, after some considerations, we chose not to do so for our first prototype because the plugin SDK imposes too many limitations on UI design. For example, it is not clear how we can create important UI components in our system such as a list with foldable items or a scrollbar with multiple colors through Photoshop plug-in SDK. Another main reason for using Connection SDK is that our standalone application can run on a variety of other devices such as tablets and smartphones to provide additional flexibility and screen space.

One implementation detail worth mentioning is that due to the latency and performance issues, it is currently not possible for Photoshop to transmit every intermediate frame to clients, especially when users issue commands in a high frequency or when the images are large. Our solution is to synchronize only the text history list when user is actively interacting with Photoshop, and sync the intermediate images in undo stack only when Photoshop is idle.

6 Evaluation

6.1 Design

We conducted a user study to evaluate the usability of our system. The study consists of three sessions: introduction, a comparison study on navigation tasks and a think-aloud exploratory session where participants freely use our system. Below are detailed descriptions of the participants and our study design.

Participants We recruited six professional artists who used Photoshop on daily basis in their works, including two professional photo retouchers, three professional photographers, and one digital image sketcher.

Introduction session Each test session started with a 5-minute introduction to our system and its features, followed by asking the participants to perform several simple trial tasks such as “fold/unfold the command group”, “perform partial comparison on eye region”, and “increase the number of semantic segments via slider”.

Comparison study session The comparison study evaluates the effectiveness of our semantic navigation function by comparing it with traditional linear text history lists and time-based clustering of command history in [Li et al. 2011] in three navigation tasks.

Each navigation task first requires the participants to watch through a pre-recorded portrait retouching session (with controllable playback speed). Then given the history list of this pre-recorded video, the participants are asked to locate approximate starting points of three semantic chunks: eye sharpening, skin smoothing, and lip retouching.

Note that we use pre-recorded history lists to facilitate the comparison between three different visualizations. The study interview confirmed that participants can all comprehend the retouching process in the video because professional artists share similar retouching techniques.

Exploratory session After the comparison study session, the participant is given an additional 20 minutes to freely explore our system with the prepared photos and graphics assets. An observer sits through the session and encourages the participant to think aloud. At the end of the session, the participant fills a 7-point Likert survey about our system features followed by an brief interview.

6.2 Results and Discussion

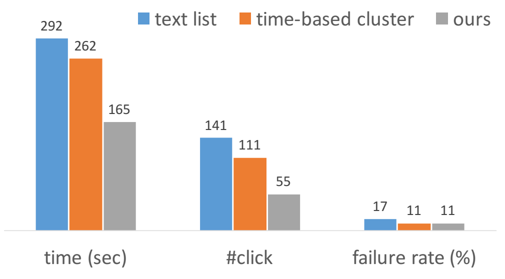


Figure 5: *Comparison study chart. We use failure-rate instead of success-rate (as in the main text) for more coherent visualization, so that all quantities are the lower the better.*

For the tasks in comparison study session, we measured the rate of success, time to completion, and the number of mouse clicks. Figure 5 showed the result of the comparison test. We also performed an ANOVA on three measured factors: time-on-task: $F = 11.83, p\text{-}level = 0.0014$; mouse click count: $F = 10.56, p\text{-}level = 0.0022$; success rate: $F = 0.75, p\text{-}level = 0.49$. The lack of sufficient differences on the success rate indicates that participants were pretty good at finding the target states using all three kinds of systems. However, participants achieve the same success rate with our system using significantly less time and fewer mouse clicks.

The post-hoc questionnaire about features of our system shows highly positive results (Figure 6). Averaged across all features, the ranking were 6.5 for both "ease of use" and "usefulness". One participant gave lower scores on semantic navigation (5) and dual view (4) features. The participant commented that his job was to retouch huge amount of photos in limited time. His per-photo history list tends to be short and thus requires little navigation. He rated highly on the command grouping function for it provides even better compact history and might lead to better efficiency.

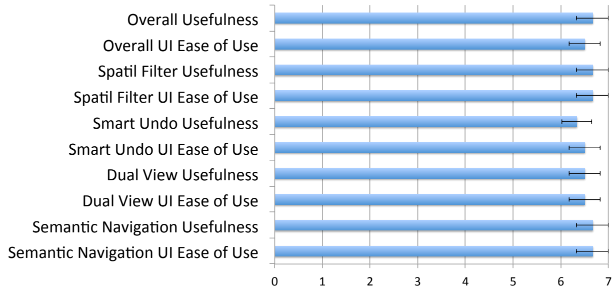


Figure 6: *Likert scale rating for importnat functions in our system. Error bars show 95% CI.*

During the final interview, all participants agreed that the scenario of rolling back to some previous states frequently happened during their daily works. They also looked forward to seeing our features appear in the Photoshop. We also received many encouraging comments from our participants. One professional photographer who uses history list and snapshot heavily for daily photo retouching works said "The history list hasn't changed too much since around Photoshop 5.0, which added the function of snapshot; your system would be really really useful for my workflow and I cannot wait to use it". The image sketcher said "I appreciate the fact that you add these cool features without increasing the size of the history list. History list review is very important to me, and although there are plug-ins available, they often took additional spaces and distracted my work". And from the professional retoucher "I mainly manage my progress via layers because the history list in Photoshop is really not that helpful. With your system, it seems like that I could finally use fewer layers in my workflow!".

7 Limitations and Future Work

Data variety We demonstrated and evaluated our UI and algorithms through only portrait retouching examples. We chose photo retouching as the primary subject of our pilot study because a) it is a common and popular type of photo editing tasks among professional photo retouchers, and b) it usually contains sufficiently long, varying, and complex editing histories to benefit from good navigation mechanisms as well as contribute to data analysis + machine learning. As a future work, we would like to extend our data collection to other image editing tasks such as scenery retouching and graphics design. Digital sketching is a particularly interesting application [Xing et al. 2014], provided future Photoshop SDK support logs of brush path and parameters.

Adaptive learning Our current implementation performs both SVM learning and n-gram analysis as an offline preprocess based on the collected data from professional artists. This fixed initialization might not adapt well to individual users or tasks with different preferences or characteristics from the initial training data. For example, users might have unique command usage patterns and the spatial elements in the feature vectors might vary for different tasks.

One interesting potential is to perform both learning processes periodically in an incremental fashion so that our system can better adapt to individual users and tasks. The n-gram analysis is already real-time as it involves only hash table lookup. But the SVM classifier would take minutes for training, which is not fast enough for online learning. We are looking at potential incremental learning algorithms as well as the possibility of using idle CPU/GPU times for such training tasks.

Nonlinear navigation Our system currently supports only linear undo, redo, and navigation. As demonstrated in earlier works such as [Chen et al. 2011; Myers et al. 2015], nonlinear explorations can provide extra flexibilities and possibilities not possible with linear exploration, such as undo a specific image region (e.g. left eye of a portrait) with histories preceding other regions that we wish to keep (e.g. right eye of the same portrait). We believe extending our current methodology from linear lists to non-linear graphs should be quite feasible from the algorithm and design perspective. However, the main issues lie in usability; non-linear histories will differ more from traditional linear histories than our linear adaptive approach. Thus, further user studies will be needed for further investigation.

Evaluation in the wild Our controlled pilot evaluation in the lab yielded encouraging and positive results as well as interesting questions. The current study used pre-recorded video for simpler between-subjects comparison and shorter study session. A natural next step would be to collect daily image editing histories from each participant and evaluate the performance on open-ended tasks. As the research community proposes different visualizations techniques, such as hierarchical graph and multi-resolution history strip, for editing history [Chen et al. 2011; Denning et al. 2011; Nancel and Cockburn 2014; Myers et al. 2015], a more thorough comparison evaluation is also required. Finally, the release of the plug-in or the integration of the proposed UI into open source image editors such as GIMP will give us more insights into how the modification on the history panel affects artists' daily workflow.

Segment labeling We are also interested in the possibility of automatic labeling the semantics segments extracted from command histories. In our preliminary experiments, we have found this a difficult task due to the high amount of ambiguity among different photo retouching techniques. For example, artists might use very similar operation sets for different tasks (e.g. retouching eyes and hairs), and they might also use very different operation sets for similar tasks (e.g. skin smoothing). We believe that with better similarity measurements (e.g. region segmentation for facial structures in [Berthouzoz et al. 2011]), it should be possible to automatically label the command histories.

8 Conclusions

In this paper we introduced a *data-driven adaptive history* that enhances the usability of traditional linear history with two main mechanisms: *semantic navigation* and *smart undo*. Our design minimally changes the look and feel of traditional history lists, while significantly enhances their usability and satisfaction for history navigation. Users can easily adapt to our UI, while the developers will find it straightforward to integrate our design into common image editing systems. Our methodology is data-driven; with proper data collection, it can be applied to different image editing tasks and systems. Our design has shown significant improvements over traditional history lists through user studies that incorporate both quantitative measurements and qualitative feedbacks. We believe that our data-driven history list could be of great benefit to digital artists, who tend to have long editing histories due to the complexity and the trial-and-error nature of their common tasks.

Acknowledgements

We would like to thank our user study participants for their time and feedbacks, and anonymous reviewers for their valuable comments. We also acknowledge Flickr user Keven Dooley for reproduced photos under Creative Commons License Attribution 2.0 Generic. This work has been partially supported by Hong Kong RGC general research fund *Workflow Textures* (project number 17202415).

References

BERGER, I., SHAMIR, A., MAHLER, M., CARTER, E., AND HODGINS, J. 2013. Style and abstraction in portrait sketching. *ACM Trans. Graph. 32*, 4 (July), 55:1–55:12.

BERTHOUZOZ, F., LI, W., DONTCHEVA, M., AND AGRAWALA, M. 2011. A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. *ACM Trans. Graph. 30*, 120:1–14.

CALLAHAN, S. P., FREIRE, J., SANTOS, E., SCHEIDEGGER, C. E., SILVA, C. T., AND VO, H. T. 2006. Vistrails: visualization meets data management. In *SIGMOD '06*, 745–747.

CHANG, C.-C., AND LIN, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2*, 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

CHAUDHURI, S., AND KOLTUN, V. 2010. Data-driven suggestions for creativity support in 3d modeling. *ACM Trans. Graph. 29*, 6 (Dec.), 183:1–183:10.

CHEN, H.-T., WEI, L.-Y., AND CHANG, C.-F. 2011. Nonlinear revision control for images. *ACM Trans. Graph. 30*, 4 (July), 105:1–105:10.

CHEN, H.-T., GROSSMAN, T., WEI, L.-Y., SCHMIDT, R. M., HARTMANN, B., FITZMAURICE, G., AND AGRAWALA, M. 2014. History assisted view authoring for 3d models. In *CHI '14*, 2027–2036.

DENNING, J. D., AND PELLACINI, F. 2013. Meshgit: Diffing and merging meshes for polygonal modeling. *ACM Trans. Graph. 32*, 4 (July), 35:1–35:10.

DENNING, J. D., KERR, W. B., AND PELLACINI, F. 2011. Meshflow: Interactive visualization of mesh construction sequences. *ACM Trans. Graph. 30*, 4 (July), 66:1–66:8.

DENNING, J. D., TIBALDO, V., AND PELLACINI, F. 2015. 3dflow: Continuous summarization of mesh editing workflows. *ACM Trans. Graph. 34*, 4 (July), 140:1–140:9.

EDWARDS, W. K., IGARASHI, T., LAMARCA, A., AND MYNATT, E. D. 2000. A temporal model for multi-level undo and redo. In *UIST '00*, 31–40.

GRABLER, F., AGRAWALA, M., LI, W., DONTCHEVA, M., AND IGARASHI, T. 2009. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph. 28*, 3 (July), 66:1–66:9.

GROSSMAN, T., MATEJKA, J., AND FITZMAURICE, G. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *UIST '10*, 143–152.

HARTMANN, B., MACDOUGALL, D., BRANDT, J., AND KLEMMER, S. R. 2010. What would other programmers do: suggesting solutions to error messages. In *CHI '10*, 1019–1028.

HEER, J., MACKINLAY, J., STOLTE, C., AND AGRAWALA, M. 2008. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics 14*, 6, 1189–1196.

IARUSSI, E., BOUSSEAU, A., AND TSANDILAS, T. 2013. The drawing assistant: Automated drawing guidance and feedback from photographs. In *UIST '13*, 183–192.

KAY, J., AND THOMAS, R. C. 1995. Studying long-term system use. *Commun. ACM 38*, 7 (July), 61–69.

KAZI, R. H., IGARASHI, T., ZHAO, S., AND DAVIS, R. 2012. Vignette: Interactive texture design and manipulation with freeform gestures for pen-and-ink illustration. In *CHI '12*, 1727–1736.

KAZI, R. H., CHEVALIER, F., GROSSMAN, T., AND FITZMAURICE, G. 2014. Kitty: Sketching dynamic and interactive illustrations. In *UIST '14*, 395–405.

KAZI, R. H., CHEVALIER, F., GROSSMAN, T., ZHAO, S., AND FITZMAURICE, G. 2014. Draco: Bringing life to illustrations with kinetic textures. In *CHI '14*, 351–360.

KELBY, S. 2011. *Professional Portrait Retouching Techniques for Photographers Using Photoshop*. Peachpit.

KURLANDER, D., AND BIER, E. A. 1988. Graphical search and replace. In *SIGGRAPH '88*, 113–120.

KURLANDER, D., AND FEINER, S. 1991. Editable graphical histories: the video. In *CHI '91*, 451–452.

KURLANDER, D., AND FEINER, S. 1992. A history-based macro by example system. In *UIST '92*, 99–106.

LAFRENIERE, B., BUNT, A., WHISSELL, J. S., CLARKE, C. L. A., AND TERRY, M. 2010. Characterizing large-scale use of a direct manipulation application in the wild. In *GI '10*, 11–18.

LAFRENIERE, B., BUNT, A., LOUNT, M., KRYNICKI, F., AND TERRY, M. A. 2011. Adaptablegimp: designing a socially-adaptable interface. In *UIST '11 Posters*, 89–90.

LI, W., MATEJKA, J., GROSSMAN, T., KONSTAN, J. A., AND FITZMAURICE, G. 2011. Design and evaluation of a command recommendation system for software applications. *ACM Trans. Comput.-Hum. Interact. 18* (July), 6:1–6:35.

LIMPAECHER, A., FELTMAN, N., TREUILLE, A., AND COHEN, M. 2013. Real-time drawing assistance through crowdsourcing. *ACM Trans. Graph. 32*, 4 (July), 54:1–54:8.

LINTON, F., AND SCHAEFER, H.-P. 2000. Recommender systems for learning: Building user and expert models through long-term observation of application use. *User Modeling and User-Adapted Interaction 10*, 2-3 (Feb.), 181–208.

LU, J., YU, F., FINKELSTEIN, A., AND DIVERDI, S. 2012. Helpinghand: example-based stroke stylization. *ACM Trans. Graph. 31*, 4 (July), 46:1–46:10.

LU, J., BARNES, C., WAN, C., ASENTE, P., MECH, R., AND FINKELSTEIN, A. 2014. Decobrush: Drawing structured decorative patterns by example. *ACM Trans. Graph. 33*, 4 (July), 90:1–90:9.

MENG, C., YASUE, M., IMAMIYA, A., AND MAO, X. 1998. Visualizing histories for selective undo and redo. In *APCHI '98*, 459.

MURPHY, G. C., KERSTEN, M., AND FINDLATER, L. 2006. How are java software developers using the eclipse ide? *IEEE Softw. 23*, 4 (July), 76–83.

MYERS, B. A., LAI, A., LE, T. M., YOON, Y., FAULRING, A., AND BRANDT, J. 2015. Selective undo support for painting applications. In *CHI '15*, 4227–4236.

NAKAMURA, T., AND IGARASHI, T. 2008. An application-independent system for visualizing user operation history. In *UIST '08*, 23–32.

NANCEL, M., AND COCKBURN, A. 2014. Causality: A conceptual model of interaction history. In *CHI '14*, 1777–1786.

PAVEL, A., BERTHOUZOZ, F., HARTMANN, B., AND AGRAWALA, M. 2013. Browsing and analyzing the command-level structure of large collections of image manipulation tutorials. Tech. Rep. UCB/EECS-2013-167, EECS Department, University of California, Berkeley, Oct.

SU, S. L., PARIS, S., AND DURAND, F. 2009. Quickselect: history-based selection expansion. In *GI '09*, 215–221.

SU, S. L. 2007. Visualizing, editing, and inferring structure in 2d graphics. In *Adjunct Proceedings of UIST*, 29–32.

XING, J., CHEN, H.-T., AND WEI, L.-Y. 2014. Autocomplete painting repetitions. *ACM Trans. Graph. 33*, 6 (Nov.), 172:1–172:11.

XING, J., WEI, L.-Y., SHIRATORI, T., AND YATANI, K. 2015. Autocomplete hand-drawn animations. *ACM Trans. Graph. 34*, 6 (Oct.), 169:1–169:11.

YU, L.-F., YEUNG, S.-K., TERZOPOULOS, D., AND CHAN, T. F. 2012. Dressup!: Outfit synthesis through automatic optimization. *ACM Trans. Graph. 31*, 6 (Nov.), 134:1–134:14.

ZITNICK, C. L. 2013. Handwriting beautification using token means. *ACM Trans. Graph. 32*, 4 (July), 53:1–53:8.

A Reference Label List

The reference label list provided to hired artists during data collection.

- global adjustment
  - color/tone adjustment
  - brightness/contrast adjustment
  - transformation (crop/rotation/perspective)
- retouching eyes
  - sharpening the eyes
  - brightening the whites of the eyes
  - removing eye veins
  - reducing dark circles under eyes
  - deforming eye
  - enhancing eyelashes
  - enhancing eyebrows
- retouching skin
  - smoothing skin
  - removing hotsopt
  - removing blemishes
  - removing wrinkle
  - applying digital makeup
- retouching hair
  - removing stray hair
  - filling hair gap
  - adding highlight to hair
- retouching lips
  - changing lip shape
  - changing lip color
  - creating glossy lip
  - whitening/repairing teeth
- slimming and trimming
  - face thinning
  - body slimming